Mining the Heterogeneous Transformations between Data Sources to Aid Record Linkage

Matthew Michelson*
Fetch Technologies
841 Apollo St., Ste. 400
El Segundo, CA 90245
mmichelson@fetch.edu

Craig A. Knoblock
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA
knoblock@isi.edu

ABSTRACT

Heterogeneous transformations are translations between strings that are not characterized by a single function. E.g., nicknames, abbreviations and synonyms are heterogeneous transformations while edit distances are not. Such transformations are useful for information retrieval, information extraction and text understanding. They are especially useful in *record linkage*, where the problem is to determine whether two records refer to the same entity by examining the similarities between their fields. However, heterogeneous transformations are usually created manually and without assurance they will be useful. This paper presents a data mining approach to discover heterogeneous transformations between two data sets, without labeled training data, which can then be used to aid record linkage. In addition to simple transformations, our algorithm finds combinatorial transformations, such as synonyms and abbreviations together. Our experiments demonstrate that our approach can discover many types of specialized transformations, and we show that by exploiting these transformations we can improve record linkage accuracy. Our approach makes discovering and exploiting heterogeneous transformations more scalable and robust by lessening the domain and human dependencies.

1. INTRODUCTION

*This research is based upon work supported in part by the Air Force Office of Scientific Research under grant number FA9550-07-1-0416, and in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-07- D-0185/0004. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Record linkage is the process of recognizing when two records refer to the same entity. This is a substantial problem when integrating multiple data sources. Record linkage is not a new problem, and has been around in various forms for a long time [4]. It sometimes goes by the names object identification [6], de-duplication [5, 10, 15] or co-reference resolution [8]. As an example, consider the two directory resources listed in Figure 1. Each data source contains a restaurant name and a manager's name, and the goal is to discover which restaurants are the same across the listings.

Figure 1: Matching Records in Two Tables

Most record linkage examines the records at the field level and then makes an overall record level decision as to whether or not the records match. In our example scenario of Figure 1, we want to make record level match decisions about restaurants based on the *manager* and *restaurant* fields. For this reason, many of the record linkage approaches use sophisticated machine learning approaches to making these record level decisions based on the field level similarities [18, 14, 2, 16]. However, until recently [18, 9], most of the methods use simple techniques such as edit distance to measure the field similarities. In the cases where record linkage does use more complex field similarity measures such as synonym comparisons [18, 9], the systems use hand coded, domain specific transformations.

In this paper, we present a method to mine complex string transformations directly from the data, which can then be used by record linkage systems. That is, given two data

sources, the algorithm presented in this paper explicitly mines various transformations that occur between the data sources (such as synonyms), without needing labeled training data. These transformations can then be used to aid record linkage by being used for field similarity measures.

In general, measuring field-level similarities between the attributes is difficult because of the myriad of possible differences in the field values. Beyond the characteristics that are easy to capture, such as spelling differences and missing or extra tokens, there are many differences that need to be captured by more specific techniques. For example, the two restaurant names of the first row of Figure 1 demonstrate the need for acronym identification since one restaurant "California Pizza Kitchen" is represented in the other set by its acronym "CPK." Another frequent field level difference that occurs is abbreviation, such as "Delicatessen" to "Deli." Yet another is a synonym/nickname relationship such as "Robert" is "Bobby" and "William" is "Bill" which is shown in Figure 1. Unlike their generic counterparts, such as edit distance, these specific field level relationships are not defined by a generic function that works in all cases across all field values. Thus, we group them all together under the heading, "heterogeneous transformations."

While work such as Minton, et. al, [9] and Tejada, Knoblock, and Minton [18] link records together based on the common heterogeneous transformations between the records, the transformations used are provided to the algorithm *a priori* and created manually. For example, in matching cars, a user might create and supply a list of synonyms such as "hatchback" equals "liftback." Beyond the cost in creating these lists, there is no assurance that the created sets of transformations will be useful for matching the records. For instance, while the list creator might think "hatchback" and "liftback" will be useful, they may only occur infrequently within the data sources or there may be cases where they are misleading.

By *mining* these transformations instead of composing them manually, we make the use of heterogeneous transformations more robust, scalable and cost effective. Further, by mining the transformations, rather than creating them, the algorithm can discover multi-token, combination transformations that would be difficult to construct manually. For instance, our algorithm discovers that "2D Coupe" and "2 Dr Hatchback" are a transformation between car trims. This transformation combines a pseudo-synonym (hatchback equals coupe), with an abbreviation (2D equals 2 Dr). Further, the algorithm selects only those mined transformations that have high mutual information, indicating that these transformations provide dependency information about their co-occurrence. In this manner, not only are the transformations created algorithmically, but they also provide a certain amount of information about whether or not they will be useful as a pair.

Although these transformations apply well to record linkage, they are not limited in their use to this application domain. Once mined, these lists of transformations could be useful for many tasks. For instance, transformations could be used in information retrieval as expanded thesauri that go beyond traditional English language thesauri. Another domain where transformations are useful is in schema mapping. For instance, giving a set of transformations to an alignment system such as iMap [3] can greatly increase its ability to find text matches within the columns which aids its alignment. Such transformations could be also used in information extraction to aid in disambiguating and discovering the extractions. For instance, by combining a set of transformations with an algorithm for resolving object resolutions [20], an extractor might be able to identify extractions it could not previously. For instance, one article might mention "CPK" and another might mention "California Pizza Kitchen," and both restaurant names could be extracted and consolidated.

The rest of this paper is organized as follows. In the next section we present the algorithm for mining heterogeneous transformations in detail. Then we present some experimental results that show we can discover these transformations and that these transformations are useful for improving record linkage. Then we present related work, and we conclude with some final thoughts.

2. MINING TRANSFORMATIONS

Our goal is to mine transformations between the data sources, without labeled data. More formally, our inputs consist of two data sets, D_i and D_j, along with a set of schema aligned attributes A_{ij}. Note that the attributes $a_i, a_j \in A_{ij}$ may be a subset of the attributes in D_i and D_j. From these inputs, we produce a set of transformations T, which relate to certain values, v, for attributes $a_{ij} \in A_{ij}$, such that the values $v(a_i) \in D_i$ can be substituted for $v(a_j) \in D_j$. For example, $v(a_i)$ may be "CPK" in D_i which can be substituted for "California Pizza Kitchen" for $v(a_j)$ in D_j. Formally, we define a transformation $t_{ij} \in T$, as a function that maps between data sources: $t_{ij}(v(a_i) \in D_i) \mapsto (v(a_j) \in D_j)$. It is important to note that the value functions v may operate on subsets of the tokens in a_i and a_j. That is, the mapping by transformation t_{ij} may map some tokens of a_i to some tokens of a_j. This definition gives us the flexibility to define heterogeneous transformations, t_{ij} to be arbitrary string substitutions, rather than strict programmatically defined functions, as with edit distances. Our hypothesis space, therefore, is the set of all possible token combinations for all possible aligned attributes values. From this space, we select the subsets of tokens that most likely aid in identifying matching records between the sources.

The formal definition of our transformation mining problem:

$$\langle D_i, D_j, A_{ij} \rangle \rightarrow \langle T \rangle$$

differentiates the problem of mining heterogeneous transformations from other data mining problems. For example, in association rule mining [1], the inputs are a single set of transactions and the output is a set of rules relating tuples in the input set of transactions. This differs from our problem, which considers two data sets from which to pull out transformations mapping values in one set to the other. Our algorithm also differs from selection algorithms. In feature selection, a set of features is either built up from the null set [7] or pruned down from the full set [11]. In either case, feature selections add or remove whole features (attributes

in our case), where as in our algorithm we explicitly build up sets of value mappings between features. In some sense, the problem of mining transformations lies between these two methods. On the one hand, we are finding mappings between attributes which is like mining associations between tuples in association rule mining. On the other hand, we are finding transformations that only apply to a subset of attributes, as in feature selection where only subsets of attributes are considered.

The overall algorithm for discovering heterogeneous transformations breaks into three high level steps, as shown in Figure 2. In the first step, we find possible matches between the sources. This is done using the cosine similarity between record pairs, which we refer to as *possible matches*. Next, we mine the transformations from these possible matches, since they give us a set of records with likely transformations contained within them. In the final step, which is optional, a user can prune incorrect transformations that the algorithm mines. Since we are mining transformations from possible matches, rather than labeled training data, errant transformations can be generated. However, we show in our experiments that pruning these incorrect transformations is optional because both the pruned and unpruned transformation sets aid the record linkage equally.

Note that although we do not require labeled training data, we do assume that the schema matching has been done. That is, we assume that each column from one source has been registered to its matching column in the other source. In our example of Figure 1 this means that the system knows that the Manager column of one source maps to the Manager column of the other source and similarly for the Restaurant columns.

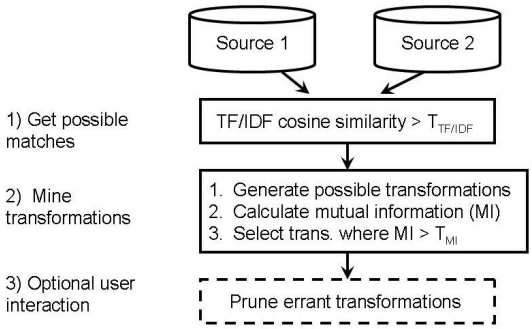

Figure 2: Algorithm for mining heterogeneous transformations

Therefore, as a first step, our algorithm must discover the possible matches between the data sources, from which we can mine the transformations. The intuition here is that likely matches between the data sources will often contain useful transformations. However, we do not want to label matches ahead of time because that will add an extra burden to the user. So instead, we introduce a threshold T_{Cos}, and we create our possible matches from record pairs between the data sources whose TF/IDF cosine similarity is above a threshold T_{Cos}. Since this threshold is chosen by a user, in our experiments we vary it and examine the results. The algorithm for generating candidate matches is given in Table 1.

Table 1: Selecting candidate matches

GetCandidateMatches(Dataset S, Dataset T)
Candidates ← {}
For all records $s \in S$ X $t \in T$
If CosineSim(s,T) > T_{Cos}
Candidates ← Candidates ∪ (s,t)

The next step is to mine the transformations from these possible matches. Intuitively, the algorithm finds sets of tokens that co-occur with each other within the possible matches, but that are not exactly the same. For instance, looking at the restaurant field of the second record in Figure 1, we see it has "Bill's" in common, but also has "Chop House" in one record and "Steak Place" in the other. If this occurs frequently in our possible matches, then this might be a transformation we would want to discover to use later for these types of restaurants. Figure 3 shows the pairs generated from the first matches shown in Figure 1. As shown, the algorithm lines up the fields across the possible matches and generates pairs of sets of tokens for all tokens in the fields that are not exactly the same.

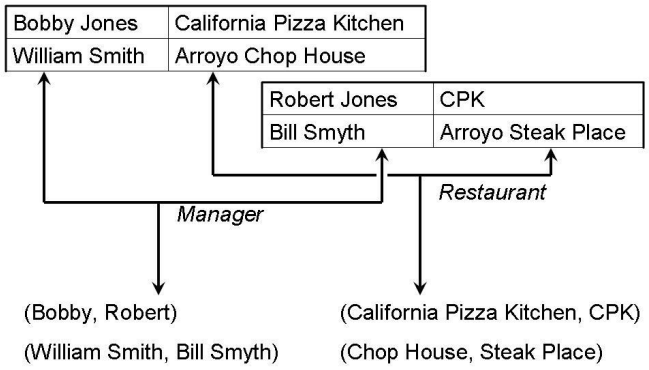

Figure 3: Generating co-occurring token pairs from possible match pairs

Of course, while this generation process creates many possible transformations, it will create both good and bad ones. Therefore, we need a method by which to select only the most promising pairs of token sets. To do this we could could consider co-occurrence in the possible matches. For example, we might use the likelihood of co-occurrence and keep only the transformations that are the most probable. However, this method does not use all of the possible information provided by the transformation pairs. A more useful metric should include not only a measure of probable co-occurrence, but also a measure of dependence between each part of the transformation. For this reason, we choose the transformations with the highest mutual information amongst the transformations mined from the possible match pairs. Those with high mutual information not only

occur with a high likelihood, but they also carry more information about whether or not the transformation occurs for that field in matches.

For this step we obtain the probabilities used in the mutual information from our set of possible matches. Given sets of tokens s and t, we define mutual information as:

$$MI(s,t) = p(s,t) * \log_2\left(\frac{p(s,t)}{p(s)p(t)}\right)$$

Once all of the co-occurring transformations in the possible matches are scored, we select only those with a mutual information above a user chosen threshold, T_{MI}. Like T_{Cos}, since T_{MI} is chosen by a user, its value is varied in the experiments to examine its behavior. The algorithm for mining the transformations from the candidate matches is shown in Table 2.

Table 2: Mining transformations

GenerateTransformations(CandidateMatches C)
Transformations $\leftarrow$ {}
For all record pairs $(s,t) \in C$
For all attributes $a \in Attributes$
$trans_{ast} \leftarrow$ NonMatchingTokens(a_s,a_t)
If MutualInformation($trans_{ast}$) $> T_{MI}$
Transformations $\leftarrow$ Transformations $\cup$ $trans_{ast}$

Note that we might wrongly exclude a transformation just because the probability of that transformation occurring is low. An example of this would be that CPK is the same as California Pizza Kitchen from Figure 3. This highlights one of the limitations of our approach, namely that reasonable but infrequent transformations will not be mined. However, infrequent transformations, such as "CPK," usually occur for acronyms, and acronyms are usually specific to a certain noun that does not necessarily reoccur in the data source. So, pruning such overly specific transformations is not an error.

There is one last step in our algorithm. Since our possible matches are generated without labeled training data, our cosine similarity method can generate noisy possible matches. Such noisy matches can produce errant transformations. For instance, consider matching two data sources of hotels, with a name, city and star rating. If there are common hotels in large cities, such as multiple Hiltons in Los Angeles, but they have a different star rating, this might lead the algorithm to produce a possibly errant transformation such as "3*" is the same as "4*." In this step, a user can choose to prune this transformation from the final set. Note that although this step might require human intervention, we feel that pruning a few errant transformations is much less costly than labeling many matches in order to mine the transformations. Further, our experiments show that when aiding record linkage, the pruned and unpruned transformation sets perform equally well. Therefore, since the pruning has little effect on how the transformations aid the record linkage, this step is optional.

3. EXPERIMENTS

In this section we present experiments that show we mine useful transformations, varying both the threshold for generating possible matches using TF/IDF (T_{Cos}) and that used to select transformations with the highest mutual information (T_{MI}). We also apply these transformations to a record linkage problem to show they help, and for the case where they do not, we argue why this is not the fault of the algorithm but a characteristic of the data to be matched. Finally, we show that pruning the errant rules has a minimal effect on the record linkage results, so this step is optional.

Our experiments focus on three sets of data sources used previously in the record linkage community. The first set of data sources, called "Cars," is used in [9] and consists of 2,777 automobile records from the Kelly Blue Book website and 3,171 records from the Edmunds car buying site. Between these sets there are 2,909 one-to-many matches. Each record in this set has a make, model, trim, and year. The next data sources, called "BFT" in [9] contain 132 hotels to be matched against 1,125 text entries from an internet bulletin board, each manually parsed into attributes. Each record in this set has a star rating, a local area and a hotel name. Between these sources there are 1,028 matches. This source is particularly noisy, containing many misspellings and missing tokens, so it is a good test for using TF/IDF to generate potential matches. In the interest of repeatability, the BFT data set is freely available from our website for experimental use.[1]

Our last data sources, called "Restaurants," consist of two tables of restaurants, which have been used in the past more than once [2, 9]. One table, with 330 records, comes from Zagats and the other contains 534 records from Fodors. These tables have 112 matches between them and each record has a name, address, city, and cuisine. Interested participants can also find this data set online.[2]

Table 3 shows some example transformations mined from each of the experimental domains. The mined transformations include synonyms, abbreviations, acronyms and combinations of these. To make the transformations easier to read, we present them as disjunctions. That is, transformations are grouped by the string from one source and we union together the strings from the other source.

One way to interpret these mined transformations are as "association rules" [1]. An association rule is of the form *antecedent* $\rightarrow$ *consequent*. In our case, we can interpret each mined transformation as a rule implying that a field from one data source can be associated with different values for that field in the other set. For instance, the transformation *Asian* $\rightarrow$ *Chinese* $\cup$ *Japanese* means that, for the matches in this set, when we see Asian for the cuisine in one record, it might refer to Japanese or Chinese in the cuisine value of its match.

Since we can consider the transformations as association rules, we can use the standard association rule metrics to evaluate the mined transformations. For these metrics, we use the true matches between the sources to see how well

[1] http://www.isi.edu/integration/people/michelso/index.html
[2] http://www.cs.utexas.edu/users/ml/riddle/data/restaurant.tar.gz

Cars Domain		
Field	*Kelly Blue Book Value*	*Edmunds Trans.*
Trim	Coupe 2D	2 Dr Hatchback
Trim	Sport Utility 4D	4 Dr 4WD SUV ∪ 4 Dr STD 4WD SUV ∪ 4 Dr SUV
BFT Domain		
Field	*Text Value*	*Hotel Trans.*
local area	DT	Downtown
local area	SD	San Diego
hotel name	Hol	Holiday
local area	Pittsburgh	PIT
Restaurants Domain		
Field	*Fodors Value*	*Zagats Trans.*
City	Los Angeles	Pasadena ∪ Studio City ∪ W. Hollywood
Cuisine	Asian	Chinese ∪ Japanese ∪ Thai ∪ Indian ∪ Seafood
Address	4th	Fourth
Name	and	&
Name	delicatessen	delis ∪ deli

Table 3: Transformations mined from different domains

our mined rules actually perform. The first metric we consider is *Support*. Support is the fraction of the matches that satisfy the transformation, out of all matches. It is defined as:

$$\text{Support} = \frac{\#\text{matches with transformations}}{\#\text{total matches}}$$

Support shows how well the transformations generalize to the true matches, in terms of their coverage. However, we also need a metric that gives a measure of how often the transformations actually apply, given the antecedent. That is, if we see the antecedent, such as Asian, how likely is it that the match will have the consequent, such as Japanese or Chinese? The metric that defines this measure is *Confidence* and it is defined as:

$$\text{Confidence} = \frac{\#\text{matches with transformations}}{\#\text{matches with antecedent}}$$

As a last metric, we consider *Lift*. Lift describes how much information the antecedent gives about the consequent for both occurring together. Therefore, Lift values above 1 are preferred. The Lift is defined as the Confidence divided by the *Expected Confidence* (EC), where EC is defined as:

$$\text{EC} = \frac{\#\text{matches with consequent}}{\#\text{total matches}}$$

Table 4 presents the association rule metrics for our mined transformations, varying the TF/IDF threshold (T_{Cos}) and the mutual information threshold (T_{MI}). For these metrics we calculate the values using all mined transformations, and we present the average.

Table 4 shows that we mine useful transformations for all of the domains, without any labeled training data. In only one case do we have an average Lift value less than 1, which means the transformations provide good information about their occurrence. Also, most of the confidence scores are high, and only a few support levels are low, and these usually occur when we could only mine a few transformations.

Cars Domain					
	T_{MI}	0.2	0.1	0.05	0.025
$T_{Cos} = 85$	Supp.	0.80	0.80	0.53	0.44
	Conf.	1.29	1.29	1.23	0.81
	Lift	1.72	1.72	3.54	0.95
	# Rules	2	2	4	19
$T_{Cos} = 65$	Supp.	0.64	0.80	0.51	0.38
	Conf.	1.31	1.29	1.27	0.78
	Lift	2.08	1.72	4.46	1.09
	# Rules	1	2	5	15
$T_{Cos} = 45$	Supp.	0.00	0.64	0.57	0.46
	Conf.	0.00	1.31	1.13	1.27
	Lift	0.00	2.08	3.93	4.51
	# Rules	0	1	3	6
BFT Domain					
	T_{MI}	0.2	0.1	0.05	0.025
$T_{Cos} = 85$	Supp.	0.14	0.13	0.13	0.13
	Conf.	0.50	0.71	0.71	0.71
	Lift	23.51	16.31	16.31	16.31
	# Rules	3	5	5	5
$T_{Cos} = 65$	Supp.	0.13	0.09	0.10	0.12
	Conf.	0.99	0.25	0.29	0.27
	Lift	48.24	21.88	12.67	9.64
	# Rules	1	10	26	41
$T_{Cos} = 45$	Supp.	0.13	0.11	0.16	0.11
	Conf.	0.99	0.59	0.33	0.28
	Lift	48.24	22.02	8.90	28.96
	# Rules	1	3	16	50
Restaurants Domain					
	T_{MI}	0.2	0.1	0.05	0.025
$T_{Cos} = 85$	Supp.	0.03	0.11	0.14	0.14
	Conf.	0.32	0.39	0.42	0.42
	Lift	29.89	9.97	7.40	7.40
	# Rules	4	13	18	18
$T_{Cos} = 65$	Supp.	0.04	0.11	0.28	0.31
	Conf.	0.71	0.54	0.66	0.61
	Lift	10.00	11.63	2.36	1.86
	# Rules	1	12	36	45
$T_{Cos} = 45$	Supp.	0.04	0.04	0.31	0.38
	Conf.	0.71	0.81	0.56	0.62
	Lift	10.00	35.00	3.91	1.56
	# Rules	1	4	44	69

Table 4: Association rule metrics for the mined transformations

Table 4 also demonstrates that a balance must be struck between the values of the metrics and the number of transformations that are mined. While the metrics may be high for certain threshold levels, the actual mined transformations may not be very useful for record linkage, especially given that they are few in number and may apply so often as to not be useful in discriminating matches from non-matches. For instance, consider the Cars domain where T_{Cos} is 0.85 and and T_{MI} is 0.1. In this case, only 2 transformations are learned, "4D" is "4 Dr" and "2D" is "2 Dr." Both of these transformations occur frequently in the matches, yielding high metrics. However, these transformations occur so frequently, across both matches and non-matches that they are not useful for discriminating between matches and non-matches. Compare these transformations to the more specific transformations shown in Table 3, which seem more useful, even though they have lower metrics. Therefore, we should not just consider the metric values, but we should

also consider the number of transformations mined.

Lastly, varying the thresholds indicates that the results seem more sensitive to T_{MI} than T_{Cos}. This is expected since T_{Cos} dictates the initial pairs we can mine from and not which transformations get selected. Note at the low values of T_{MI} we mine many more transformations and the metrics only decrease slightly. This is better behavior for record linkage where we want to mine many transformations, with most of them useful rather than just a few. The results indicate that for the high level of T_{Cos} we stagnate in mining transformations across the values of T_{MI}, since we have many fewer record pairs to mine from, yielding just a few repeated transformations.

In general, it seems the best way to select the thresholds is to set T_{Cos} such that it does not limit the transformations that could be mined, and to use a low value of T_{MI} to make sure the algorithm selects a fair number of possible transformations. For this reason, in our record linkage results below we use the transformations mined with T_{Cos} of 0.65 and T_{MI} of 0.025. These threshold yield a large number of transformations with good metrics, and should therefore be useful to aid the record linkage. As we show below, even though these low thresholds yield some noisy transformations, these do not affect the record linkage results.

For our record linkage experiments, we use a copy of the HFM record linkage system [9] to which we supply the mined transformations. However, unlike in that paper, due to implementation issues we could not use Support Vector Machines to make the record level match decisions. Instead, we use C4.5 decision trees [13]. We compare HFM using our mined special transformations along with its usual transformations (Equals, Levenshtein distance, Prefix, Suffix, Concatenation, Abbreviation and Missing) to HFM using just its usual transformations alone, without our mined transformations. We also compare using the full set of mined transformations to the set of user-pruned transformations.

To do the pruning, we remove all transformations that are incorrect. In the Cars domain, we removed only 1 transformation out of 8, "wagon → sedan." For the Restaurants domain we prune 6 out of the 26 mined transformations. These often come from the address field and seem to be specific to certain record pairs only, suggesting that they slip in under the T_{MI} threshold. For example, we prune "2nd at 10th st. → second." Lastly, in the BFT domain we prune the most transformations, 28 out of 40. Nine of these 28 are the case described in Section 2, where hotel names and locations are similar, but the star ratings are not, producing transformations such as "3* → 4* ∪ 2* ∪ 2.5*." A similar case occurs 13 times, where a rare area and star rating are the same but the hotel name is not, resulting in transformations such as "Marriott → Coronado Del Hotel."

The record linkage results are shown in Table 5. For the record linkage setting, we follow most record linkage papers [2, 9] and use 2 fold cross validation. This means we label 50% of the data for training and test on the remaining 50%. We do this across 10 trials and present the average values.

Note that across all domains, the precision and recall differences using the full set of transformations versus the pruned set are not statistically significant using a two-tailed t-test with α=0.05. Therefore, they are effectually the same, so pruning the transformations becomes an optional step since there is no difference in the record linkage utility. Even in the BFT domain, where we pruned 28 transformations, the decision tree learned in record linkage ignores most of the incorrect transformations while using the correct ones common to both the pruned and unpruned sets.

Cars Domain		
	Recall	Precision
No trans.	66.75	84.74
Full Trans.	**75.12**	83.73
Pruned Trans.	75.12	83.73
BFT Domain		
	Recall	Precision
No trans.	79.17	93.82
Full Trans.	**82.89**	92.56
Pruned Trans.	82.47	92.87
Restaurants Domain		
	Recall	Precision
No trans.	91.00	97.05
Full Trans.	91.01	97.79
Pruned Trans.	90.83	97.79

Table 5: Record linkage results both using and not using the mined transformations

For the Cars and BFT domain, we see a statistically significant increase in the recall, while the differences in the precisions are not statistically significant using a two-tailed t-test with α=0.05. (Note that the F-measures are also statistically significant.) An increase in recall, without any change to precision, means that record linkage is able to discover new matches, without harming its ability to classify the matches it already can find. In the Cars domain, this translates into 115 more matches using the transformations, and in the BFT domain this represents 23 more matches. The increase in recall is lower in the BFT domain than the Cars domain because the noisy nature of the data not only makes it difficult to mine the transformations, but applying them is difficult as well, since a mined transformation might not apply in the many misspelled cases. Nonetheless, even on noisy data, we are able to improve the record linkage process.

For the Restaurant domain, neither the differences in recall nor precision are statistically significant when we include the transformations versus not, using a two-tailed t-test with α=0.05. This was surprising given that this domain yielded some of the most interesting mined transformation. The explanation for this can be found by looking at the record linkage process. In this domain the transformations are often not used because the attributes to which they apply are not used for deciding matches. In fact, in this domain many of the transformations apply to the cuisine field, but the decision tree, which makes accurate record level decisions, almost exclusively relies on the name and address field. So the cuisine field is not needed to make correct matches since the name and address are sufficient. Therefore, for the mined

transformations to be useful they must also apply to attributes that are useful for deciding matches. Even if the transformations are extremely useful in terms of support and confidence, they will be ignored if the attribute they apply to is not needed. Lastly, we would like to note that these record linkage results have as much to do with the HFM system as the mined transformations, which is why we emphasize the association rule metrics. Perhaps another record linkage system, using the transformations differently, could improve the record linkage results even more.

4. RELATED WORK

As stated previously, we can view our mined transformations as association rules [1]. In our case, the value for an attribute from one source is the antecedent and the values it transforms into in the other source is the consequent. In fact, there has even been work on mining association rules using mutual information [17]. However, the problem domain is different between mining association rules and mining transformations. Association rules come from a set of transactions. For instance, given a set of users and what they purchase at the grocery store, an association rule might be "people who bought cereal also bought milk." In this case, there is only one data set, and the goal is to find the links from any subset of transactions to another. When mining transformations, our task is to take a set of possibly matching record pairs and within these find transformations that will help in deciding matches during record linkage.

The use of word and phrase co-occurrence to find similar words or phrases has been done extensively in natural language processing (NLP). For example, Turney [19] uses word co-occurrence based on information retrieval results to define sets of synonyms. More recently, there has been work that takes this idea further to identify paraphrases and generate grammatical sentences by looking at co-occurring sets of words [12]. The major difference between the work in NLP and our work is that we do not focus on language, and as such, we are not limited to word based transformations such as substitutions and synonyms. The transformation that "4D" is "4 Dr" is not really lexical, but we can still exploit co-occurrence to discover such heterogeneous transformations. Our method allows us to extend the set of heterogeneous transformations we can learn using co-occurrence because we are not constrained by the need to use real language, we only use the "language" set by the data sources.

This work is also somewhat similar to the Resolver system [20] in that Resolver is able to consolidate synonymous mentions on the Web, such as "Red Planet" is the same as "Mars." However, the input to their system is a set of relations (extractions), such as $(lacks, Mars, ozone\ layer)$ and $(lacks, Red\ Planet, ozone\ layer)$ from which they consolidate synonyms. Our input, in contrast, are two sets of data sources, from which we mine transformations from between the possible matches. Again, this means we often find non-linguistic transformations, such as "4D" is "4 Dr" whereas their synonyms will be for nouns, since they are extractions from language on the Web. Note, however, that as stated in the introduction, by combining our work with a powerful consolidation algorithm like Resolver, we may be able to perform even more accurate extractions.

5. CONCLUSION

In this paper we presented an algorithm for mining heterogeneous transformations from data sources without labeled matches between the sources. Although the transformations could be applied in other application domains, such as text understanding and information retrieval, these transformations are particularly useful for record linkage. We first find a set of possible matches based on the cosine similarity between record pairs, and then we mine transformations with the highest mutual information amongst these pairs.

One interesting conclusion we draw from this work is that there are actually features of the data sets that determine whether or not the mined transformations are useful, independently of the mined transformations. In particular, even if we mine the most useful transformations, if the attributes they apply to are not used to determine record level matches they will ultimately be ignored. For instance, in the Restaurants domain we find that while we learn interesting transformations for the cuisine field, this field is not needed to make record level decisions since the name and address fields can be used almost exclusively. In the future we will investigate methods to determine whether or not it is worth using the mined transformations by looking directly at the data and concluding if the attributes will be useful for record linkage.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of ACM SIGKDD-03*, pages 39 – 48. ACM Press, 2003.

[3] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: Discovering complex semantic matches between database schemas. In *Proceedings of SIGMOD*, 2004.

[4] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.

[5] M. A. Hernandez and S. J. Stolfo. The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD Conference*, 1995.

[6] T. Huang and S. J. Russell. Object identification in a bayesian context. In *IJCAI-97*, pages 1276–1283, 1997.

[7] K. Kira and L. A. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proceedings of AAAI*, pages 129–134, 1992.

[8] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *Neural Information Processing Systems (NIPS)*, 2004.

[9] S. N. Minton, C. Nanjo, C. A. Knoblock, M. Michalowski, and M. Michelson. A heterogeneous field matching method for record linkage. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM-05)*, 2005.

[10] A. E. Monge and C. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of ACM SIGKDD-96*, pages 267–270, 1996.

[11] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature selection. *IEEE Transactions on Computers*, C-26(9):917–922, 1977.

[12] B. Pang, K. Knight, and D. Marcu. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of HLT-NAACL*, 2003.

[13] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[14] P. Ravikumar and W. W. Cohen. A hierarchical graphical model for record linkage. In *UAI 2004*, 2004.

[15] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of ACM SIGKDD-02*, 2002.

[16] P. Singla and P. Domingos. Entity resolution with markov logic. In *Proceedings of the 6th IEEE International Conference on Data Mining*, 2006.

[17] B. K. Sy. *Machine Learning and Data Mining in Pattern Recognition*, chapter Discovering Association Patterns Based on Mutual Information, pages 369–378. Springer Berlin / Heidelberg, 2003.

[18] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of ACM SIGKDD-02*, 2002.

[19] P. D. Turney. Mining the Web for synonyms: PMI–IR versus LSA on TOEFL. *Lecture Notes in Computer Science*, 2167:491–503, 2001.

[20] A. Yates and O. Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of NAACL-HLT*, 2007.